

| Nro. ord. | Apellido y nombre | L.U. | #hojas |
|-----------|-------------------|------|--------|
|           |                   |      |        |

# ORGANIZACIÓN DEL COMPUTADOR I - Parcial

Verano 2016

| Ej.1 | Ej.2 | Ej.3 | Ej.4 | Ej.5 | Nota |
|------|------|------|------|------|------|
|      |      |      |      |      |      |

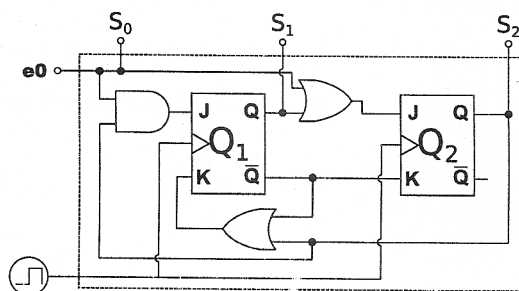
Corrector:

## Aclaraciones

- Anote apellido, nombre, LU y numere *todas* las hojas entregadas
- Cada ejercicio se califica con Bien, Regular o Mal. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se califican globalmente.
- El parcial **NO** es a libro abierto, pero puede tener los apuntes provistos por la cátedra y dos hojas A4 con apuntes propios
- **Importante:** Justifique sus respuestas. Las soluciones a ejercicios de la práctica que se utilicen deben ser incluidas en el examen.
- Son suficientes para aprobar:
  - 4 ejercicios Bien y ninguno Mal.
  - Los ejercicios 1, 2 y 3 Bien y no más de un ejercicio Mal.

## Ejercicio 1

- a. Analizar el comportamiento del siguiente circuito y expresarlo en una tabla:



- b. Asumiendo que  $e_0$  siempre vale 0, mostrar la transición de estados que se produce en la memoria interna. Realizar lo mismo asumiendo que vale 1. Para cada una de las dos configuraciones: definir si existen o no estados estables (una configuración es estable si al llegar a ella el próximo valor es sí mismo). En caso de que existan expresarlo, caso contrario justificar por qué.
- c. Extender el circuito anterior con un circuito que tome las salidas del anterior y tenga 3 salidas en la que se expresen:
- Si el numeral  $[s_2s_1s_0]$ , interpretado en complemento a 2 de 3 bits es el máximo representable
  - Si el numeral  $[s_2s_1s_0]$ , interpretado en complemento a 2 de 3 bits es el mínimo representable
  - Si la cantidad de bits que valen cero es par

**Ejercicio 2** Se tiene una máquina de acumulador –tiene un único registro de propósito general AC– llamada JULIES3. La JULIES3 tiene como unidad direccionable el *nibble* (4 bits) y las palabras son de 12 bits. Cuenta con el siguiente conjunto de instrucciones:

| opCode | Instrucción | Descripción   |
|--------|-------------|---|
| 0001   | LOAD X      | Carga el contenido de la dirección X en AC                              |
| 0010   | STORE X     | Guarda el contenido de AC en la dirección X                             |
| 0011   | ADD X       | Suma el contenido de la dirección X a AC y guarda el resultado en AC    |
| 0100   | SUBT X      | Resta el contenido de la dirección X a AC y almacena el resultado en AC |
| 0101   | INPUT       | Lee un valor del teclado y lo guarda en AC                              |
| 0110   | OUTPUT      | Escribe el valor de AC en la pantalla                                   |
| 0111   | HALT        | Termina la ejecución  |
| 1000   | SKIPCOND C  | Saltea la próxima instrucción si se cumple la condición C               |
| 1001   | JUMP X      | Carga el valor X en el PC   |
| 1010   | XORB        | Realiza un XOR entre bits de AC (ver detalle)                           |

Donde X es una dirección de memoria –al ser el único modo de direccionamiento no utiliza la sintaxis [ ] como su equivalente en la ORGA1– y C una constante de 2 bits (ver descripción a

continuación). La instrucción XORB realiza la operación XOR entre distintos pares de *bits* de AC. El efecto obtenido en el *i*-ésimo *bit* de AC es:

$$AC_i \leftarrow \begin{cases} AC_i \oplus AC_{i-1} & \text{si } i \text{ es impar} \\ AC_{i+1} \oplus AC_i & \text{sino} \end{cases}$$

Las instrucciones son de largo fijo de 12 *bits*, con el siguiente formato:

|        |    |   |   |           |   |   |   |   |   |   |   |
|--------|----|---|---|-----------|---|---|---|---|---|---|---|
| 11     | 10 | 9 | 8 | 7         | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| opCode |    |   |   | Dirección |   |   |   |   |   |   |   |

Las instrucciones que no utilizan una dirección dejan los *bits*  $b_7 \dots b_0$  en cero.

El valor de C (usado en SKIPCOND) se codifica en los *bits* 7 y 6 (los demás *bits* permanecen en cero):

- 00 verifica si  $AC < 0$
- 01 verifica si  $AC = 0$
- 10 verifica si  $AC > 0$
- 11 condición siempre verdadera

Se quiere ensamblar y cargar desde la posición de memoria 0x60 el siguiente código en una JULIES3:

```

main:  LOAD 0xFE
      ADD 0x32
      OUTPUT
      INPUT
      SKIPCOND 10
print: OUTPUT
      SKIPCOND 01
again: JUMP main
end:   HALT

```

- a. Definir a qué posición de memoria corresponde cada etiqueta.
- b. Indicar el contenido de la memoria luego de ensamblar y cargar el código anterior.
- c. Para cada parte de la planilla de seguimiento de la máquina ORGA1, justificar si es necesaria o no dicha celda. Indicar si es necesario agregar nuevas celdas o no para poder realizar el seguimiento.
- d. Modificar la planilla de seguimiento de la ORGA1 y realizar el seguimiento de la JULIES3 con la memoria presentada a continuación hasta decidir si se detiene o no (justificar):

|       |       |      |       |       |       |       |       |       |       |
|-------|-------|------|-------|-------|-------|-------|-------|-------|-------|
| 0xA0  | 0xA3  | 0xA6 | 0xA9  | 0xAC  | 0xAF  | 0xB2  | 0xB5  | 0xB8  | 0xBB  |
| 0x1B8 | 0x3B2 | 0x80 | 0x9A3 | 0x2B5 | 0x8C0 | 0xDEC | 0xAAA | 0x914 | 0x000 |

El PC inicia en 0xA0, el AC en 0x000 y la memoria toda en cero salvo las posiciones indicadas.

- e. Definir:
  - I. El tamaño máximo de la memoria
  - II. El tamaño del PC
  - III. La cantidad de instrucciones sin operandos que podrían agregarse a este formato de instrucción.
  - IV. Si la cantidad de memoria que tiene una JULIES3 es 64 *bytes*, ¿Afecta esto a las instrucciones? En caso de que no justificar por qué, en caso de que sí dar un ejemplo de dónde es que afecta el normal funcionamiento de la JULIES3.

### Ejercicio 3

- a. Realice el diagrama del *datapath* de una microarquitectura para la JULIES3 que soporte la ejecución de las instrucciones descritas (excepto HALT, INPUT y OUTPUT). Recuerde indicar el tamaño de cada registro, de los buses internos y externos, las señales de cada componente y justificar la utilización de cada componente escogido y cada decisión tomada.

Para realizar el *datapath* puede utilizar los siguientes componentes:

- una única ALU que realiza las operaciones suma y resta, sin flags.
- un único incrementador, cuyo única operación es sumar uno.

Al incluirlos detallar cuidadosamente el tamaño de los registros y los nombres de las señales. Cualquier otro componente a utilizar deberá ser implementado e incluido en el examen.

- Escriba las microinstrucciones que debe ejecutar la máquina para realizar el *fetch* de una instrucción (no incluir etapas posteriores del ciclo de instrucción).
- Escriba el microprograma que realiza la parte de ejecución del ciclo de instrucción de las siguientes instrucciones:
  - ADD 0x35
  - SKIPCOND 01
  - XORB

**Ejercicio 4** Una computadora ORGA1i mantiene la presión de salida de una máquina de espuma dentro de cierto rango. Existe un sensor de presión de la espuma que tiene mapeado un registro de E/S en la dirección 0xFFF0. Este registro informa la presión actual (PRESION\_STATUS). En R1 se mantiene la cuenta de cuantas veces se verificó la presión desde la última falla. La máquina ejecuta el siguiente programa:

```

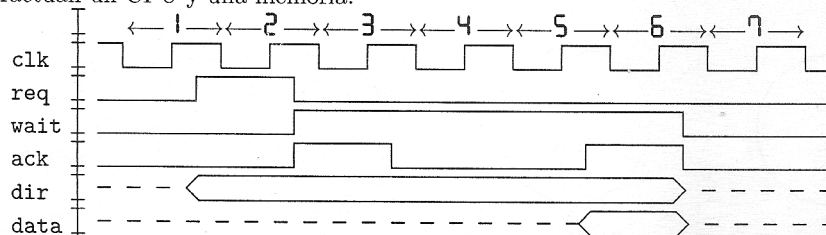
verifPresion:  MOV R0, [0xFFF0]
                CMP R0, 0x0100
                JG alarmaMuchaPr
                CMP R0, 0x0050
                JL alarmaPocaPr
                ADD R1, 0x0001
                JMP verifPresion
alarmaMuchaPr:  MOV R0, 0xFAFA
                CALL changeP
                JMP verifPresion
alarmaPocaPr:  MOV R0, 0x6868
                CALL changeP
                JMP verifPresion
changeP:        MOV [0xFFF1], R0
                MOV R1, 0x0000
                RET
  
```

Además se sabe que el ciclo de instrucción de las instrucciones de este programa toma:

- 3ms para las comparaciones
- 6ms para los movimientos de datos que involucren a la memoria
- 2ms para los saltos incondicionales
- para todas las demás toma 2ms

- Suponiendo que la presión nunca sale de rango, ¿cuántos ms se demora cada ciclo que verifica el estado de la presión? ¿cuál es la frecuencia en Hz del ciclo? (Recuerde que  $1\text{ Hz} = \frac{1}{\text{seg}}$ )
- Si el sensor de presión solicitara una interrupción cuando la presión supere el valor máximo, modifique el código presentado para aprovechar esta funcionalidad. Recalcular la frecuencia del punto anterior con la nueva configuración.
- Otro modelo de sensor cuenta con una señal que se dispara cuando la presión está fuera de rango. ¿Cuál sería la rutina de atención a esta interrupción? ¿Cómo impacta este nuevo modelo en el código original? ¿En qué contexto conviene tener un dispositivo como éste? ¿Cuál sería la nueva frecuencia de muestreo?

**Ejercicio 5** Observando un *bus* se obtiene el siguiente diagrama de tiempos donde se sabe que interactúan un CPU y una memoria:



Definir para cada una de las siguientes máquinas de estado si es compatible o no con el diagrama. Justificar.

